Description

RANDOMIZED MODULAR REDUCTION METHOD AND HARDWARE THEREFOR

TECHNICAL FIELD

The invention relates to arithmetic processing and calculating, especially for use in cryptography applications. The invention relates in particular to residue arithmetic involving modular reduction, especially computations derived from the Barrett reduction method.

BACKGROUND ART

Numerous cryptographic algorithms make use of large-integer multiplication (or exponentiation) and reduction of the product to a residue value that is congruent for a specified modulus that is related to the cryptographic key. Such computations may be susceptible to power analysis and timing attacks. Therefore, it is important that computations be secured so that information about the key cannot be obtained.

At the same time, it is important that these computations be fast and accurate. The large integer multiplication and reduction is usually the most computationally intensive portion of a cryptographic algorithm. Several distinct computational techniques have been developed for efficient modular reduction, including those known as the Quisquater method, the Barrett method and the Montgomery method, along with modifications involving precomputation and table look-up. These well-known techniques are described and compared in the prior art. See, for example: (1) A. Bosselaers et al., "Comparison of three modular reduction functions", Advances in Cryptology/ Crypto '93, LNCS 773, Springer-Verlag, 1994, pp. 175-186. (2) Jean François Dhem,

"Design of an efficient public-key cryptographic library for RISC-based smart cards", doctoral dissertation, Université catholique de Louvain, Louvain-la-Neuve, Belgium, May 1998. (3) C. H. Lim et al., "Fast Modular Reduction With Precomputation", preprint, 1999 (available from CiteSeer Scientific Literature Digital Library, citeseer.nj.nec.com/109504.html). (4) Hollmann et al., "Method and Device for Executing a Decrypting Mechanism through Calculating a Standardized Modular Exponentiation for Thwarting Timing Attacks", United States Patent No. 6,366,673 B1, Apr. 2, 2002 (based on application filed Sept. 15, 1998).

An objective of the present invention is to provide an improvement of the Barrett modular reduction method and computing apparatus therefor, which is more secure against cryptoanalysis attacks, while still providing fast and accurate results.

Another objective of the present invention is to provide the aforementioned improved method and apparatus which speeds up quotient estimation.

DISCLOSURE OF THE INVENTION

These objects are met by a computer-implemented modular reduction method in which a quotient used for the computation is systematically underestimated with a randomized error of a few bits, e.g., less than one-half word. The resulting remainder is always congruent to the corresponding intermediate product relative to the specified modulus, but is larger than the residue value and differs in a random manner for each execution. Because the quotient needs only be approximated, its estimation is faster. Because the estimation error is deliberately randomized, the method is more secure against cryptoanalysis. Yet the intermediate results are mathematically equivalent (congruent to the true results), and the final result (after a final set of

subtractions by the modulus) is the exactly the same, thus achieving the accuracy needed for the invertibility of cryptographic operations.

The hardware used to execute the method steps of the invention includes a random number generator to inject random error into the quotient estimation. A computation unit with memory access and carry injection operates under the control of an operation sequencer executing firmware to carry out the word-wide multiply-accumulate steps of large integer multiplication and modular reduction.

BRIEF DESCRIPTION OF THE DRAWING

Fig. 1 is a schematic plan view of computational hardware in accord with the present invention (including a random number generator unit), which is used to execute the modular reduction method of the present invention.

Fig. 2 is a flow diagram illustrating the general steps in the present modular reduction method.

BEST MODE OF CARRYING OUT THE INVENTION

With reference to Fig. 1, computational hardware includes a computation unit 10 that is able to perform word-wide multiply and multiply-accumulate steps on operands retrieved from memory (RAM) 12 and carry terms from registers 14. An operation sequencer 16 comprises logic circuitry for controlling the computation unit 10 in accord with firmware or software instructions for the set of operations to carry out the large-integer multiplication (or exponentiation) and the modular reduction. The operation parameters, stored in registers 18 accessible by the operation sequencer 16, consist in pointers that enable the operation sequencer to locate an operand within the RAM 12, as well as information about the lengths (number of words) of the operands, carry

injection control information, and the destination address of the intermediate results. So far, the apparatus is substantially similar to other available hardware adapted for large-integer arithmetic operations. Other than the details of the reduction steps, which will be described below, the firmware or software instructions are also similar to prior programs for executing efficient large-integer multiplication or exponentiation in word-wide segments.

Unlike prior hardware of this type, the hardware in Fig. 1 also includes a random number generator 20, which for example can be any known pseudo-random number generator circuit. The random number generator performs a calculation and outputs a random number used in the present method. Here, the random number generator 20 is accessed by the computation unit 10, as directed by the operation sequencer 16 in accord with the program instructions implementing the method of the present invention, in order to inject the randomized error quantity into the quotient estimation, as described below.

With reference to Fig. 2, the method of the present invention is an improvement of the Barrett modular reduction technique, providing faster quotient estimation and resistance to cryptoanalytic attack. The method is executed by the hardware in Fig. 1.

Modular reduction generally solves $R \equiv X \bmod M \equiv X - \lfloor X/M \rfloor M$, where R is the residue value to be found which is congruent to X for modulus M, and the symbol $\lfloor a \rfloor$ represents the floor function (the largest integer $\leq a$) so that $\lfloor X/M \rfloor$ corresponds to an integer division. The number X to be reduced is typically a product of two large (usually prime) integers, $X = A \cdot B$, i.e., where one or both of the integers A and B are of multi-word size (e.g., A and B might be 1024 bits each,

i.e. 32 32-bit words long). In any case, the basic problem in any modular reduction method is in evaluating the quotient $q = \lfloor X/M \rfloor$ in an efficient way for large (multi-word) numbers X and M.. In the present invention, an additional problem is in performing the reduction in a way that is secure from power analysis attacks in cryptographic applications.

Barrett's method involves precalculating and storing a scaled estimate of the modulus' reciprocal, U, and replacing the long division with multiplications and word shifts (dividing by b) in order to estimate the quotient. With appropriate choice of parameters, the error in the quotient estimate is at most two. The present invention improves upon Barrett's method by only approximating the quotient with a less precise but faster estimation, and by intentionally injecting a random error into the quotient prior to computing the remainder. The resulting remainder will be slightly larger than, but congruent with, the residue value.

Let w represent the word size (e.g., w = 32 for 32-bit processors), $b = 2^w$ represent the radix, n be the length of the modulus M in words, where

$$M = \Sigma_{i=0}^{n-1} m_i b^i,$$

$$0 < m_{n-1} < b,$$

$$0 \leq m_i < b, \text{ for } i = 0 \text{ to } n-2,$$

$$b^{n-1} \leq M < b^n,$$

and X be the number to be reduced, which is up to 2n +1 words in length, i.e., where

$$X = \Sigma_{i=0}^{2n} x_i b^i,$$

$$0 \leq x_i < b, \text{ for } i = 0 \text{ to } 2n,$$

$$0 \leq X < b^{2n+1} \text{ (or } M \leq X < b^{2n+1} \text{ in certain circumstances)}$$

We begin by precomputing and storing (step 30 in Fig. 2) a constant U representing the scaled reciprocal of the modulus M

$$U = \lfloor b^{2n+1}/M \rfloor = \lfloor 2^{2nw+w}/M \rfloor$$

This stored value is then subsequently used in all reduction operations for this particular modulus M. U is always n+1 words long for every modulus M which is not a power of b.

To perform a modulo reduction of X, we estimate a quotient q (step 32) using the stored value U:

$$q = \lfloor \left(\lfloor X/b^{n} \rfloor \cdot U \right) /b^{n+2} \rfloor = \lfloor \left(\lfloor X/2^{nw} \rfloor \cdot U \right) /2^{nw+2w} \rfloor$$

This requires only multiplications and word-size shifts for the computation. The floor functions tend to ensure that the quotient q is consistently underestimated (never overestimated) although it is possible that the quotient estimate will happen to be exact. A supplemental subtraction by one could be included if underestimation is required. Both the constant U and the quotient estimation differ from that of Barrett by an extra shift each of one word. (Barrett uses $U = \lfloor b^{2n}/M \rfloor$ and $q = \lfloor \left(\lfloor X/b^{n-1} \rfloor \cdot U \right) /b^{n+1} \rfloor$.) The estimated quotient $q \geq 0$ will be a maximum of n+1 words long.

At this stage, it is preferable to inject (step 36) a random error E into the computed quotient to obtain a randomized quotient, $q' = q - E$. In this case, we must have $M \cdot 2^{w/2} \leq X < b^{2n+1}$ to avoid having negative numbers. The random error E may be generated (step 34) by any known random or pseudo-random number generator (hardware or software). The only constraint is that the error fall within a specified range, such as

$$0 \leq E < (2^{w/2} - 1)$$

This limits the potential error contributed by the random generator to a specified number of bits, e.g. half a word, in addition to any error arising from the quotient estimation itself.

Next, we compute (step 38) the remainder R′, which will be congruent (modulo M) with the residue value R:

$$R' = X - q'M$$

Because the quotient q is underestimated, and a random error E is introduced, the remainder $R' \geq R$, i.e. the calculated remainder will be larger than or equal to the residue by some small random multiple of the modulus M.

The randomized remainder R′ can be used in further calculations (step 48), such as multiply or add, with another remainder R″ (randomized or not), which if necessary is again reduced (returning to step 32) for consistency. (The error remains bounded.)

Alternatively, if randomization is not required, we can choose to keep the near quotient q (step 44). In this case, we can have $0 \leq X \leq b^{2n+1}$. Keeping near the quotient will permit one to get the true remainder (steps 46 and 40).

Finally, depending upon the needs of the particular application, the residue R can be calculated from the remainder R′ by applying subtractions of the modulus M (step 40) until the number is smaller than M. The residue value R which equals R′ after the final subtraction can then be returned for use in the remainder of the cryptography system (step 42).

Randomizing the modular reduction provides security against various cryptoanalytic attacks that rely

upon consistency in power usage to determine the modulus. Here, the reduction of X modulo M varies randomly from one execution to the next, while still producing an intermediate remainder R′ that is congruent. The number of subtractions at the end to generate a final residue value R also varies randomly from one execution to the next. The number X to be reduced in this way can be obtained from a variety of different arithmetic operations, including multiplication, squaring, exponentiation, addition, etc. Likewise, the modulus M to be used can be derived in a variety of ways, most usually in cryptography from a key. The randomized modular reduction method of the present invention is useful in many cryptographic algorithms that rely upon such reduction, including both large prime (e.g. RSA) and elliptic curve-based public-key cryptography systems.